Polygonal Meshes

Adam Finkelstein & Tim Weyrich
Princeton University
COS 426, Spring 2008

3D Object Representations

Points
- o Range image
- o Point cloud

Surfaces
- o Polygonal mesh
- o Subdivision
- o Parametric
- o Implicit

Solids
- o Voxels
- o BSP tree
- o CSG
- o Sweep

High-level structures
- o Scene graph
- o Application specific

3D Object Representations

Points
- o Range image
- o Point cloud

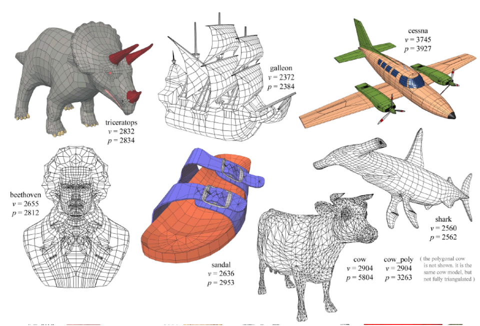
Surfaces
- ➢ Polygonal mesh
- o Subdivision
- o Parametric
- o Implicit

Solids
- o Voxels
- o BSP tree
- o CSG
- o Sweep

High-level structures
- o Scene graph
- o Application specific

3D Polygonal Mesh

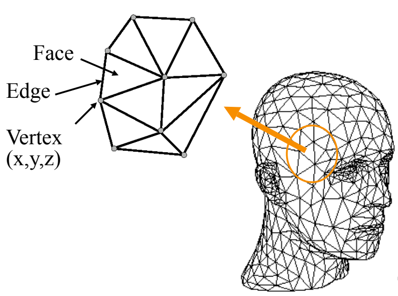
Set of polygons representing a 2D surface embedded in 3D



Isenberg

3D Polygonal Mesh

Geometry & topology



Zorin & Schroeder

Geometry background

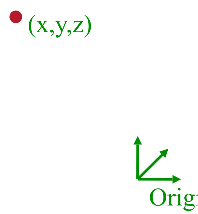
Scene is usually approximated by 3D primitives
- o Point
- o Vector
- o Line segment
- o Ray
- o Line
- o Plane
- o Polygon

3D Point

Specifies a location

- o Represented by three coordinates
- o Infinitely small

```
typedef struct {
    Coordinate x;
    Coordinate y;
    Coordinate z;
} Point;
```

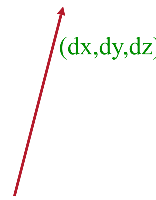


3D Vector

Specifies a direction and a magnitude

- o Represented by three coordinates
- o Magnitude $||V|| = \text{sqrt}(dx\, dx + dy\, dy + dz\, dz)$
- o Has no location

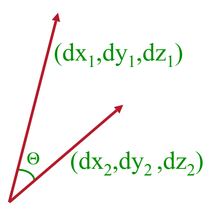
```
typedef struct {
    Coordinate dx;
    Coordinate dy;
    Coordinate dz;
} Vector;
```



3D Vector

Dot product of two 3D vectors

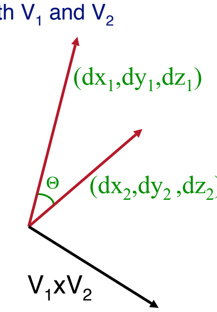
- o $V_1 \cdot V_2 = ||V_1||\ ||V_2|| \cos(\Theta)$



3D Vector

Cross product of two 3D vectors

- o $V_1 x V_2 = (dy_1dx_2 - dz_1dy_2, dz_1dx_2 - dx_1dz_2, dx_1dy_2 - dy_1dx_2)$
- o $V_1 x V_2$ = vector perpendicular to both V_1 and V_2
- o $||V_1 x V_2|| = ||V_1||\ ||V_2|| \sin(\Theta)$

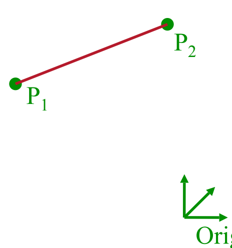


3D Line Segment

Linear path between two points

- o Parametric representation:
  - » $P = P_1 + t\,(P_2 - P_1)$, $(0 \le t \le 1)$

```
typedef struct {
    Point P1;
    Point P2;
} Segment;
```

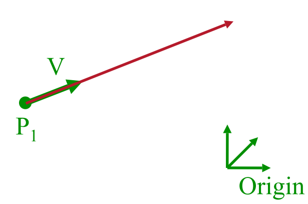


3D Ray

Line segment with one endpoint at infinity

- o Parametric representation:
  - » $P = P_1 + t\,V$, $(0 <= t < \infty)$

```
typedef struct {
    Point P1;
    Vector V;
} Ray;
```

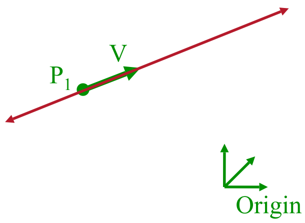


3D Line

Line segment with both endpoints at infinity

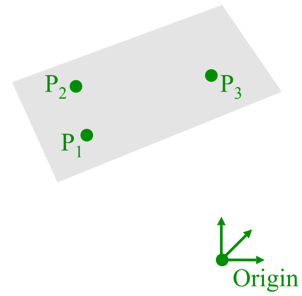
- Parametric representation:
  - » $P = P_1 + t\,V, \quad (-\infty < t < \infty)$

```
typedef struct {
    Point P1;
    Vector V;
} Line;
```



3D Plane

A linear combination of three points



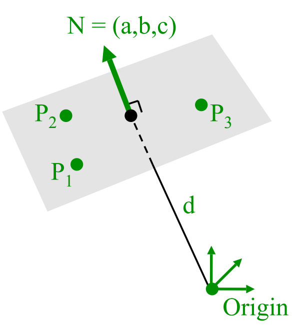
3D Plane

A linear combination of three points

- Implicit representation:
  - » $P \cdot N + d = 0$, or
  - » $ax + by + cz + d = 0$

```
typedef struct {
    Vector N;
    Distance d;
} Plane;
```

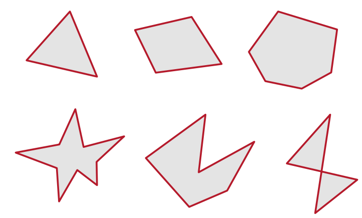
- N is the plane "normal"
  - » Unit-length vector
  - » Perpendicular to plane



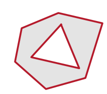
3D Polygon

Set of points "inside" a sequence of coplanar points

```
typedef struct {
    Point *points;
    int npoints;
} Polygon;
```

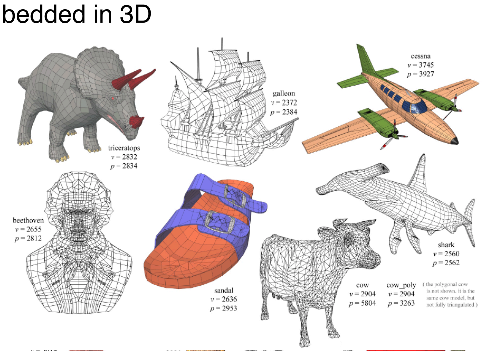


Points are in counter-clockwise order



3D Polygonal Mesh

Set of polygons representing a 2D surface embedded in 3D

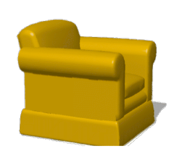


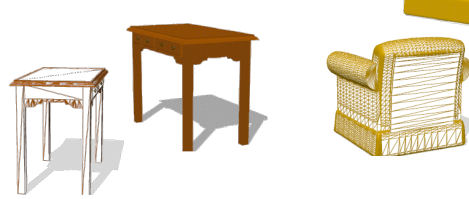
Isenberg

3D Polygonal Meshes

Why are they of interest?

- Simple, common representation
- Rendering with hardware support
- Output of many acquisition tools
- Input to many simulation/analysis tools





Viewpoint

3D Polygonal Meshes

Properties
- + Efficient display
- + Easy acquisition
- – Accurate
- – Concise
- – Intuitive editing
- – Efficient editing
- – Efficient intersections
- – Guaranteed validity
- – Guaranteed smoothness
- – etc.

NVIDIA 9600 GT GPU

Outline

Acquisition ←

Processing

Representation

Polygonal Mesh Acquisition

Interactive modeling
- o Polygon editors
- o Interchange formats

Scanners
- o Laser range scanners
- o Geological survey
- o CAT, MRI, etc. (isosurfaces)

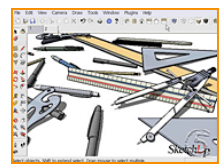
Simulations
- o Physical processes

Polygonal Mesh Acquisition

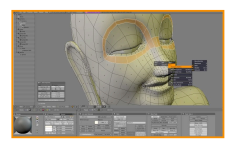
Interactive modeling
- ➢ Polygon editors
- o Interchange formats

Scanners
- o Laser range scanners
- o Geological survey
- o CAT, MRI, etc. (isosurfaces)

Simulations
- o Physical processes



Sketchup



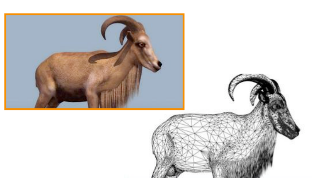
Blender

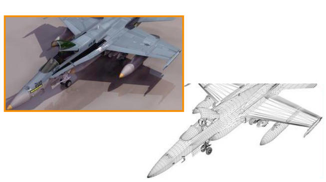
Polygonal Mesh Acquisition

Interactive modeling
- o Polygon editors
- ➢ Interchange formats

Scanners
- o Laser range scanners
- o Geological survey
- o CAT, MRI, etc.

Simulations
- o Physical processes





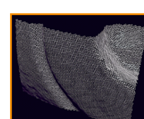
Jose Maria De Espona

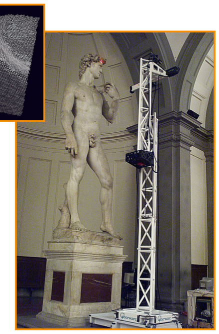
Polygonal Mesh Acquisition

Interactive modeling
- o Polygon editors
- o Interchange formats

Scanners
- ➢ Laser range scanners
- o Geological survey
- o CAT, MRI, etc. (isosurfaces)

Simulations
- o Physical processes





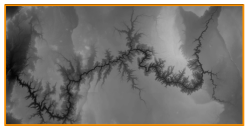
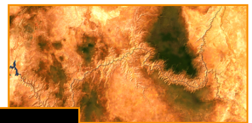
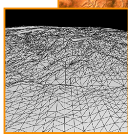
Digital Michelangelo Project
Stanford

Polygonal Mesh Acquisition

Interactive modeling
- o Polygon editors
- o Interchange formats

Scanners
- o Laser range scanners
- ➢ Geological survey
- o CAT, MRI, etc. (isosurfaces)

Simulations
- o Physical processes

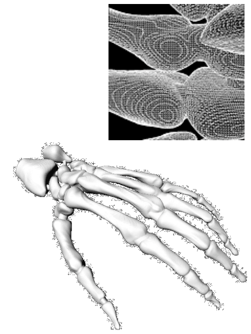
Large Geometric Model Repository Georgia Tech

Polygonal Mesh Acquisition

Interactive modeling
- o Polygon editors
- o Interchange formats

Scanners
- o Laser range scanners
- o Geological survey
- ➢ CAT, MRI, etc. (isosurfaces)

Simulations
- o Physical processes



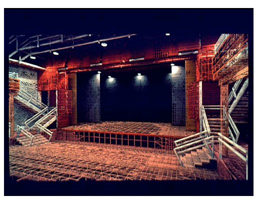
Large Geometric Model Repository Georgia Tech

Polygonal Mesh Acquisition

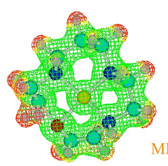
Interactive modeling
- o Polygon editors
- o Interchange formats

Scanners
- o Laser range scanners
- o Geological survey
- o CAT, MRI, etc. (isosurfaces)

Simulations
- ➢ Physical processes



SGI



MIT

Outline

Acquisition

Processing ←

Representation

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

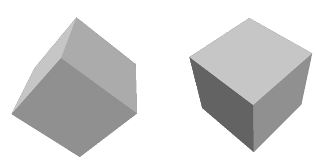
Analysis
- o Normals
- o Curvature

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature

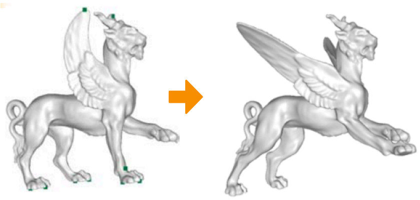



Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



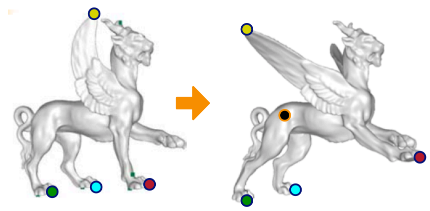
Sheffer

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



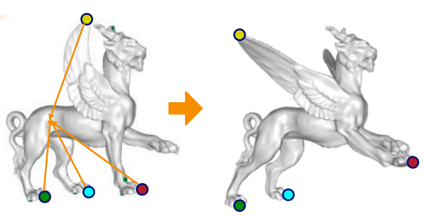
Sheffer

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



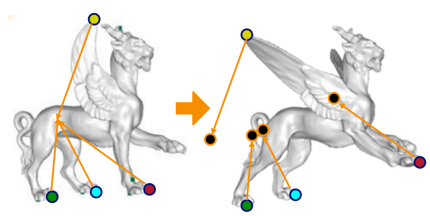
Sheffer

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



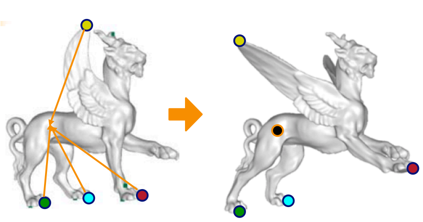
Sheffer

Polygonal Mesh Processing

Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



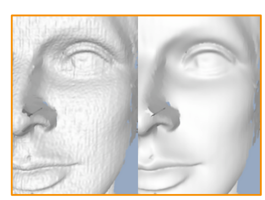
Sheffer

Polygonal Mesh Processing

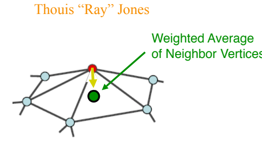
Warps
- o Rotate
- o Deform

Filters
- o Smooth
- o Sharpen
- o Truncate
- o Bevel

Analysis
- o Normals
- o Curvature



Thouis “Ray” Jones



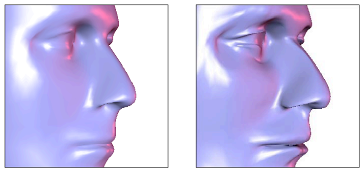
Olga Sorkine

Polygonal Mesh Processing

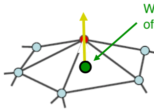
Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature



Desbrun



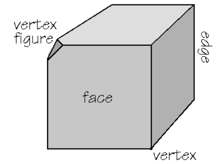
Olga Sorkine

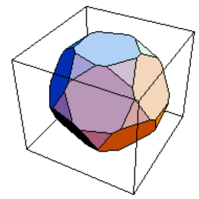
Polygonal Mesh Processing

Warps
- Rotate
- Deform

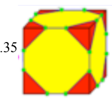
Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature





0.35



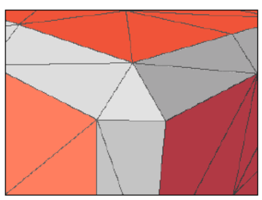
Conway

Polygonal Mesh Processing

Warps
- Rotate
- Deform

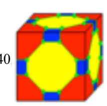
Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature



Jarek Rossignac

0.40



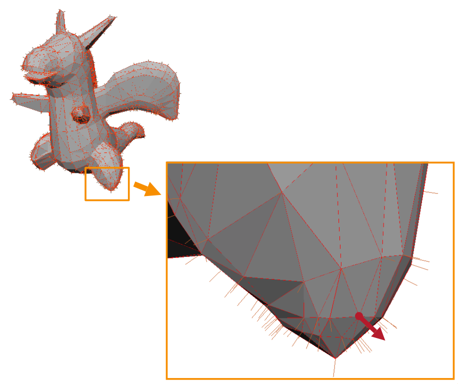
Conway

Polygonal Mesh Processing

Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature



Polygonal Mesh Processing

Warps
- Rotate
- Deform

Filters
- Smooth
- Sharpen
- Truncate
- Bevel

Analysis
- Normals
- Curvature

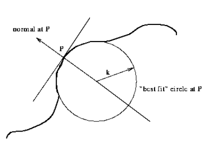


Figure 32: curvature of curve at P is $1/k$

Szymon Rusinkiewicz

Polygonal Mesh Processing

Remeshing
- Subdivide
- Resample
- Simplify

Topological fixup
- Fill holes
- Fix cracks
- Fix self-intersections

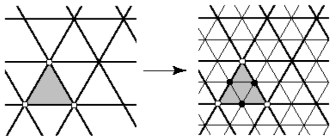
Boolean operations
- Crop
- Subtract

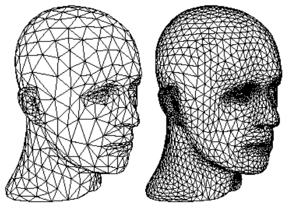
Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract





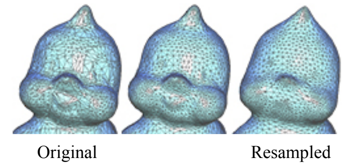
Zorin & Schroeder

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



Sorkine

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract

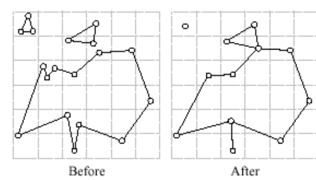
Garland

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



Vertex Clustering

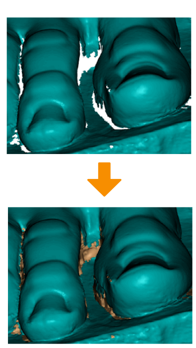
Rossignac

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



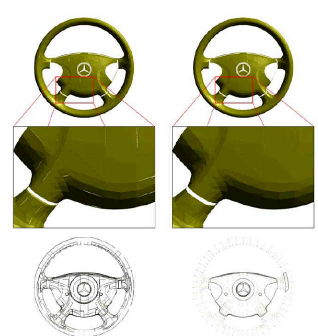
Podolak

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



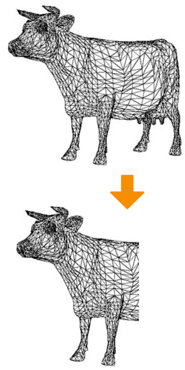
Borodin

Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract

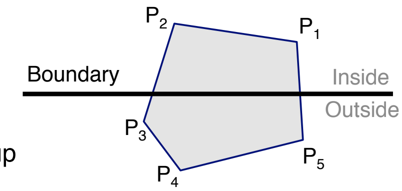


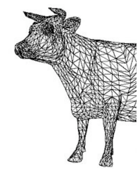
Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



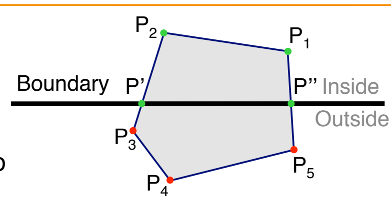


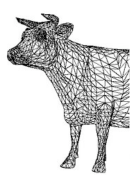
Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



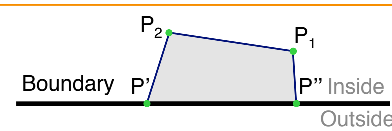


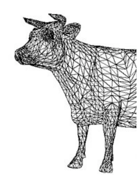
Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



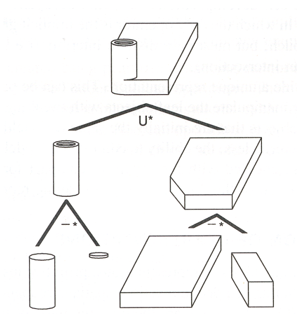


Polygonal Mesh Processing

Remeshing
- o Subdivide
- o Resample
- o Simplify

Topological fixup
- o Fill holes
- o Fix cracks
- o Fix self-intersections

Boolean operations
- o Crop
- o Subtract



FvDFH Figure 12.27

Polygonal Mesh Processing

Procedural generation
- o Surface of revolution
- o Sweep
- o Fractalize

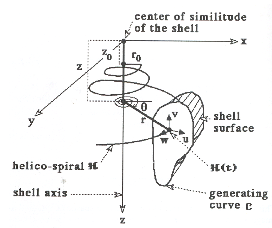
Polygonal Mesh Processing

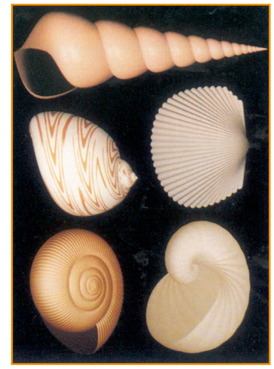
Procedural generation
- o Surface of revolution
- o Sweep
- o Fractalize

Blinn

Polygonal Mesh Processing

Procedural generation
- o Surface of revolution
- ➢ Sweep
- o Fractalize

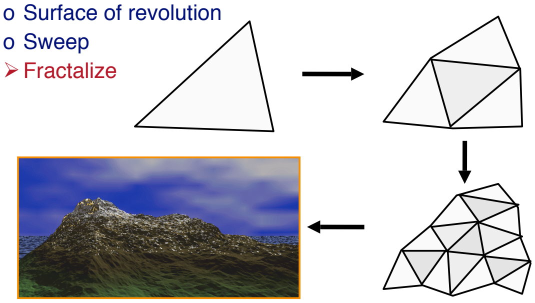




Fowler

Polygonal Mesh Processing

Procedural generation
- o Surface of revolution
- o Sweep
- ➢ Fractalize



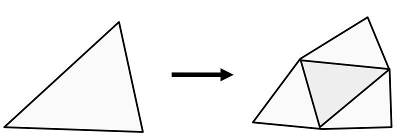
Dirk Balfanz, Igor Guskov,
Sanjeev Kumar, & Rudro Samanta,

Polygonal Mesh Processing

Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex

Polygonal Mesh Processing

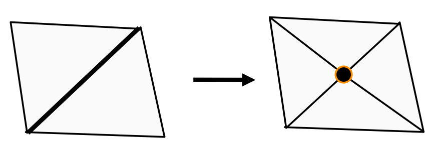
Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex



Subdivide face

Polygonal Mesh Processing

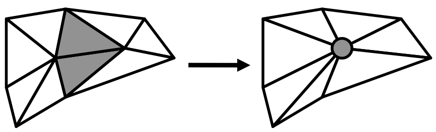
Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex



Subdivide edge

Polygonal Mesh Processing

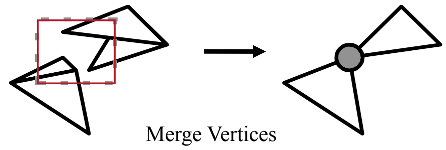
Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex



Collapse edge

Polygonal Mesh Processing

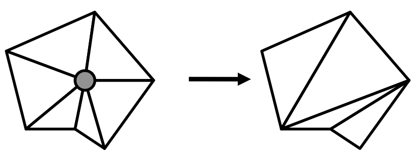
Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex



Merge Vertices

Polygonal Mesh Processing

Most operations use a few low-level operations:
- o Subdivide face
- o Subdivide edge
- o Collapse edge
- o Merge vertices
- o Remove vertex



Remove Vertex

Outline

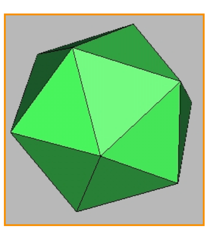
Acquisition

Processing

Representation ←

Polygon Mesh Representation

Data structures determine algorithms
- o Data structure must support key operations of algorithm efficiently

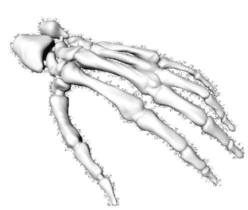


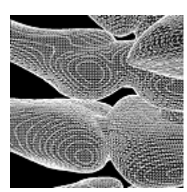
Examples:
- o Drawing a mesh
- o Removing a vertex
- o Smoothing a region
- o Intersecting polyhedra

Different data structures for different algorithms

Polygon Mesh Representation

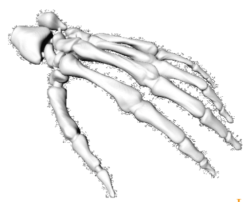
Important properties of mesh representation?

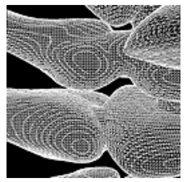




Polygon Mesh Representation

Important properties of mesh representation?
- o Efficient traversal of topology
- o Efficient use of memory
- o Efficient updates

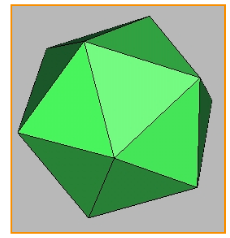




Large Geometric Model Repository
Georgia Tech

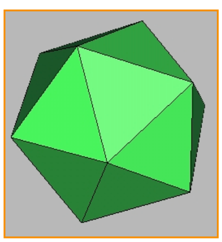
Polygon Mesh Representation

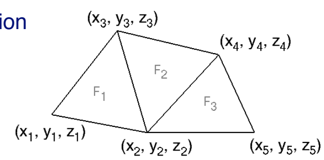
Possible data structures
- o List of independent faces
- o Vertex and face tables
- o Adjacency lists
- o Winged edge
- o Half edge
- o etc.



Independent Faces

Each face lists vertex coordinates
- o Redundant vertices
- o No adjacency information

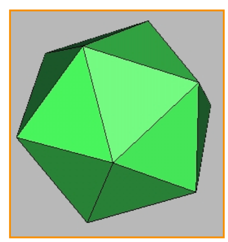


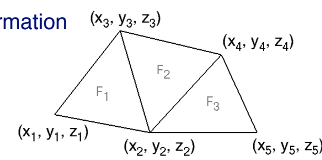


FACE TABLE	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)

Vertex and Face Tables

Each face lists vertex references
- o Shared vertices
- o Still no adjacency information



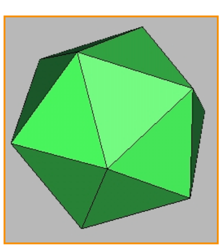


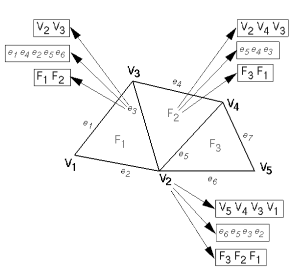
VERTEX TABLE			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

FACE TABLE			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4

Adjacency Lists

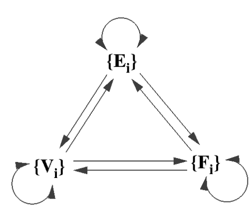
Store all vertex, edge, and face adjacencies
- o Efficient adjacency traversal
- o Extra storage

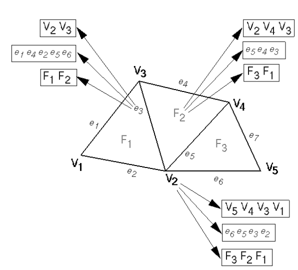




Partial Adjacency Lists

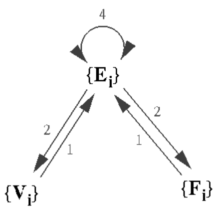
Can we store only some adjacency relationships and derive others?

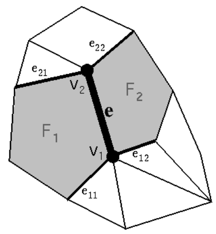




Winged Edge

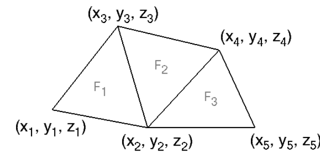
Adjacency encoded in edges
- All adjacencies in O(1) time
- Little extra storage (fixed records)
- Arbitrary polygons





Winged Edge

Example:



VERTEX TABLE

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

EDGE TABLE

					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	V3	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

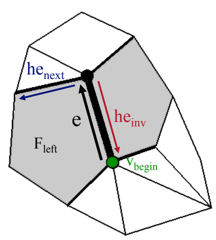
FACE TABLE

F_1	e_1
F_2	e_3
F_3	e_5

Half Edge

Adjacency encoded in edges
- All adjacencies in O(1) time
- Little extra storage (fixed records)
- Arbitrary polygons

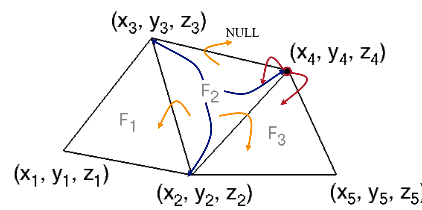
Similar to winged-edge, except adjacency encoded in half-edges



Simple Triangle Mesh

Do not store edges at all
- All faces have 3 vertices and 3 neighbors

Store adjacency in vertices and faces
- For each face: 3 vertices and 3 faces
- For each vertex: N faces



Summary

Polygonal meshes
- Easy acquisition
- Fast rendering

Processing operations
- Must consider irregular vertex sampling
- Must handle/avoid topological degeneracies

Representation
- Which adjacency relationships to store depend on which operations must be efficient